Apprentissage du langage java et développement d'application Android

Partie 2

Table des matières:

Introduction

Grace à cette partie numéro deux, vous apprendrez les bases du développement d'application Android. Nous commencerons, tout d'abord par le téléchargement et l'installation des outils qui nous aiderons à développer une application.

Ensuite, nous aborderons aussi la classe View, les widgets, les Intents et la création d'interface. Nous finirons sur l'internationalisation d'une application pour la rendre compréhensible par la majorité des utilisateurs de Smartphone Android.

Bonne lecture !

SDK Android

Google nous fournit leSDK (Software Development Toolkit) car exploiter une nouvelle plateforme n'est pas chose aisée. Ce SDK est un ensemble d'outils qui permet au développeur de créer des applications.

Le Sdk est composé de plusieurs éléments:

- Des API (interface de programmation)
  Une API c'est un ensemble de classes regroupant des fonctions mises à disposition des développeurs. Dans la majorité des cas les API effectuent des traitements de bas niveau et proposent au développeur une interface de plus haut niveau pour lui simplifier l'utilisation de fonctionnalités. Par exemple, une API graphique permet d'afficher des éléments graphiques tels que des boutons, textbox, etc…, sans avoir à gérer le périphérique dans son intégralité.

- Des exemples de code
- De la documentation
- Des outils:

  - DDMS: C'est un outil d'assistance technique

  - Mksdcard : Sert à créer des cartes mémoires logicielles utilisables avec l'émulateur.

  - Hierarchyviewer: Visualisation de la hiérarchie du layout.

  - sqlite3 : Permet d'accéder aux fichiers de donnée au format SQLite.

- Un émulateur
  L'émulateur est autonome et peut être exécuté dans le but de donner aux utilisateurs et aux développeurs la possibilité d'interagir avec le système d'exploitation Android disponible sur téléphones.

Outils du SDK en détail

DDMS

L'outil appelé DDMS(Dalvic Debug Monitor Service) est un vrai couteau suisse qui nous permet par exemple de parcourir les fichier journaux (Log), de modifier la position GPS fournie par l'émulateur, de simuler des appels ou la réception de messages et de parcourir le contenu de la mémoire interne de l'émulateur pour pouvoir y placer ou extraire des fichiers.

Le module ADT nous permet d'accéder au DDMS directement depuis éclipse. Nous pouvons aussi faire ce que fait le DDMS depuis éclipse (avec une belle interface) via un Shell et Telnet par ligne de commande. En voici un exemple ci-dessous:

1. Création d'un fichier .bat avec ce code:

```
cd C:\Android\android-sdk_r06-windows\android-sdk-windows\tools\
cmd
pause
cls
```

2. On lui dit de se connecter en Telnet à notre émulateur:

   telnet localhost 5554 [1] => Enter

   Ce message apparaît:

   Android Console: type 'help' for a list of commands OK

3. Pour simuler un appel, on tape la commande:

   gsm call 1208 [2] *=> Enter*

4. Pour simuler la réception d'un sms:

   sms send 1208 hello [3] *=> Enter*

[1] 5554 : le numéro local de l'émulateur, affiché en haut de la fenêtre de l'émulateur.
[2] 1208 : c'est le numéro qui nous appelle.
[3] hello : texte du message reçu.

Debug:

Le module ADT permet aussi le débogage de l'application depuis éclipse.

Si on veut lancer un débogage, il faut lancer l'application en mode Debug et si ce n'est pas fait automatiquement, aller sur la page Debug.

Pour déboguer le code on utilise des Breakpoints (points d'arrêt) comme avec tout autre logiciel de programmation. Cela nous permet d'exécuter l'application en mode pas-à-pas, afin de suivre précisément le cheminement de l'exécution de l'application.

Touche à utiliser pour le mode pas-à-pas:

F5: exécution du mode pas-à-pas en entrant dans les méthodes appelées dans le code.

F6: exécution pas-à-pas sans les méthodes exécutée dans le code.

F7: exécution pas-à-pas en sortant de la méthode actuelle.

F8: continuer l'exécution en sortant du mode pas-à-pas.

Hierarchyviewer

Cet outil permet de visualiser une hiérarchie. Conçu pour nous aider à consulter nos Layouts tels qu'ils sont vus par une activité en cours d'exécution dans l'émulateur.

Emulateur

Les personnes qui ne possèdent pas de Smartphone ont à disposition un émulateur fournis par le SDK Android. Celui-ci émule différents périphériques disponibles sur les vrais téléphones (par exemple un trackball, une batterie, l'accéléromètre) et permet d'exécuter le système Android et de pouvoir déployer et exécuter l'application que nous développons.

Voici quelques raccourcis clavier qui pourraient nous servir:

L'émulateur utilise un certain nombre de raccourcis clavier à connaitre:
Les flèches curseur: contrôlent le pavé/joystick du téléphone
ESC: équivaut à la touche BACK du téléphone
Home: équivaut à la touche HOME du téléphone
F2: équivaut à la touche MENU du téléphone
F3: équivaut à la touche verte Appeler du téléphone
F4: équivaut à la touche rouge Raccrocher du téléphone
F5: équivaut à la touche Search du téléphone
F7: équivaut à la touche Power du téléphone
CTRL+F3: active la caméra
CTRL+F5: augmente le volume

CTRL+F6: diminue le volume
F8: active ou désactive la simulation du réseau cellulaire
Alt+Entrée: bascule en plein écran.
CTRL+F12: bascule du mode portrait en mode paysage et inversement.

Eclipse HELIOS:

Le logiciel ECLIPSE est un environnement intégré de développement (IDE) pour beaucoup de langages, nous l'utiliserons pour coder en JAVA. Ce logiciel est gratuit.

La spécificité d'Eclipse vient du fait que son architecture est totalement développée autour de la notion de plugin.

«Architecture du logiciel Eclipse:

La base de cet environnement de développement intégré est l'*Eclipse Platform*, composée de:

- *Platform Runtime* démarrant la plateforme et gérant les plug-ins
- SWT, la bibliothèque graphique de base de l'EDI
- JFace, une bibliothèque graphique de plus haut niveau basée sur SWT
- *Eclipse Workbench*, la dernière couche graphique permettant de manipuler des composants, tels que des vues, des éditeurs et des perspectives.

Ces composants de base peuvent être réutilisés pour développer des clients lourds indépendants d'Eclipse grâce au projet Eclipse RCP (*Rich Client Platform*).

L'ensemble des outils de développement Java sont ensuite ajoutés en tant que plugins regroupés dans le projet *Java Development Tools* (JDT). Ces plugins sont architecturés selon les recommandations de OSGI.»

ADT Plugin Eclipse:

Android Development Tools (ADT) est un plugin pour le programme Eclipse. Il est conçu pour nous donner un puissant environnement pour la création d'applications Android.

ADT s'emploie à étendre les capacités d'Eclipse pour permettre de rapidement mettre en place de nouveaux projets Android, créer une interface d'application, ajouter des composants basés sur l'API Android, déboguer les applications en utilisant les outils du SDK Android, et même l'exportation signée (ou non signée) d'APKs afin de distribuer l'application sur «l'Android market».

Développement dans Eclipse avec ADT est fortement recommandé par les

développeurs expérimentés et est le moyen le plus rapide pour commencer sous Windows. Avec la mise en place de projets très simples, tels que l'intégration d'outils, d'éditeurs XML et déboguer, ADT devrait nous donner un bon coup de pouce pour le développement d'applications Android.

JDK:

Java Development Kit permet de **programmer des applications en Java**, une plate-forme de développement compatible avec Mac, Linux et Windows. Le langage créé par Sun Microsystems est apprécié notamment pour cette polyvalence.

Ce kit de développement comprend plusieurs outils de programmation dont un **compilateur et un débogueur**.

Il **inclut également l'environnement d'exécution Java** (JRE), au cas où il ferait défaut sur votre machine.

Java Development Kit regroupe les outils indispensables pour les programmeurs qui veulent s'essayer au langage Java.

Framework des méthodes de saisie

Android 1.5 a introduit le Framework des méthodes de saisies nommé IMF (Input Method Framework) que l'on peut appeler clavier virtuel ou clavier logiciel.

Certain Smartphone tel que le HTC Désire n'ont pas de clavier physique; d'autres comme le Désire Z ont un clavier physique mais pas disponible en permanence (clavier à ouvrir si on veut l'utiliser).L'IMF sait gérer tous ces scénarios, il fera apparaître un clavier virtuel dès l'absence de clavier physique.

Installation

Pour commencer l'installation, rendez-vous sur l'adresse internet suivante:

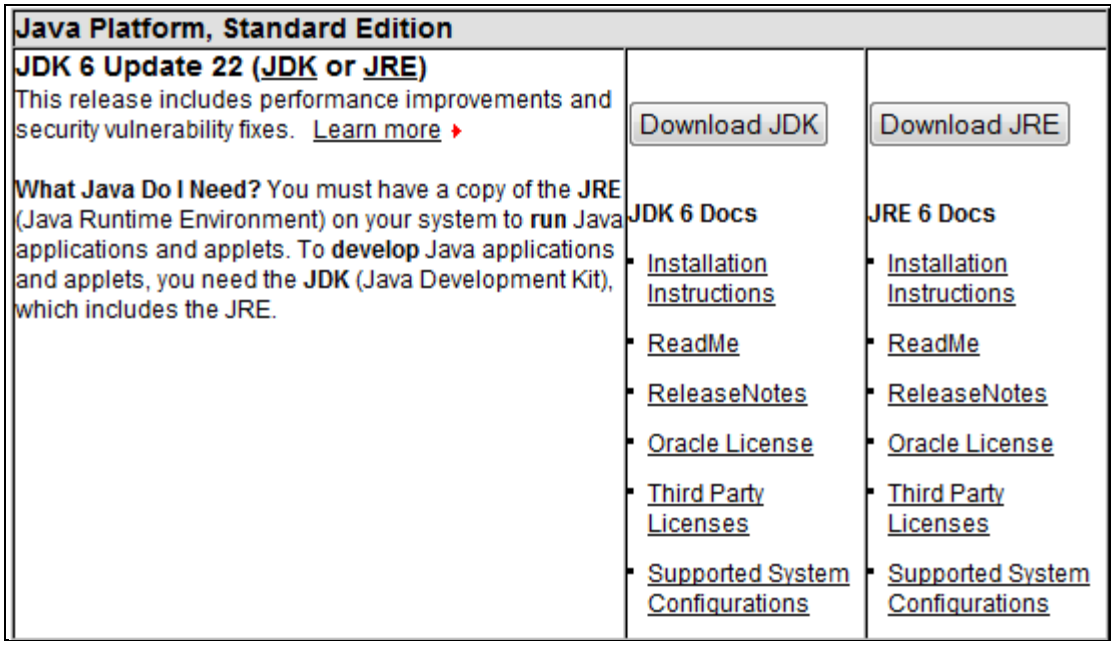
- http://developer.android.com/sdk/index.html
- Rubrique Quick Start

On y trouvera les liens suivant:

- Hardware et logiciel requis pour que le SDK fonctionne correctement.
- Lien permettant de télécharger le plugin ADT.
- Lien permettant de télécharger le logiciel Eclipse Hélios.
- Lien permettant de télécharger le **JDK** (Java Development Kit)

Installation du JDK:

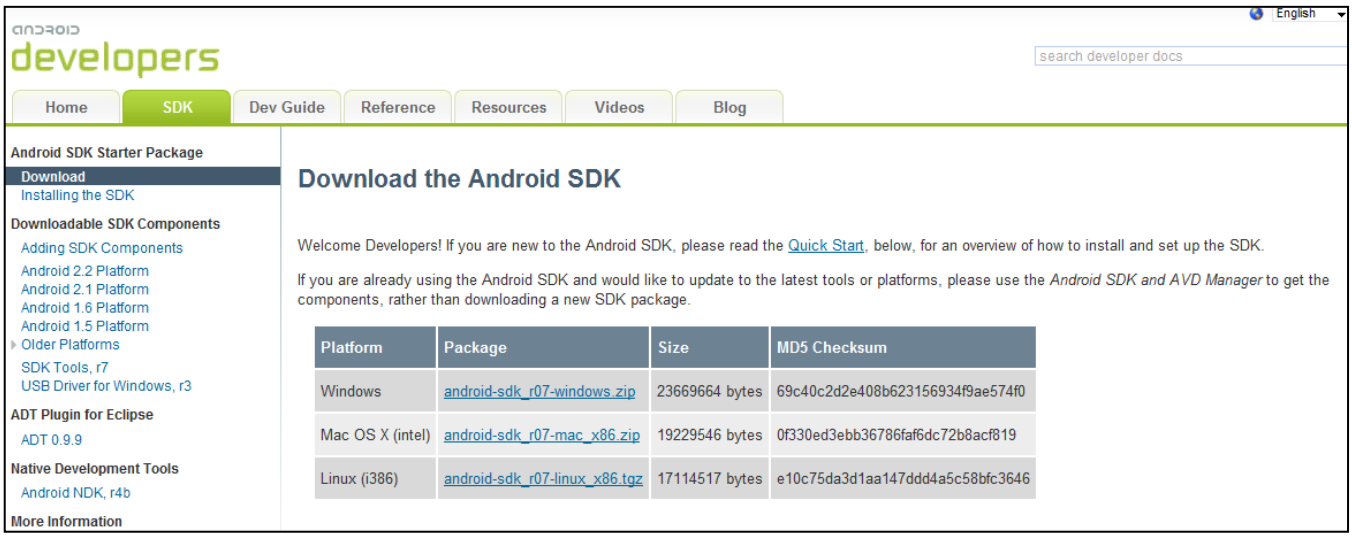
- Téléchargez le JDK correspondant à votre système d'exploitation sur votre ordinateur.



- Lancez l'installation grâce au fichier «.exe» préalablement téléchargé, puis suivez les instructions données.

Installation du SDK Android:

- Télécharger le sdk correspondant à votre système d'exploitation à cette adresse: http://developer.android.com/sdk/index.html

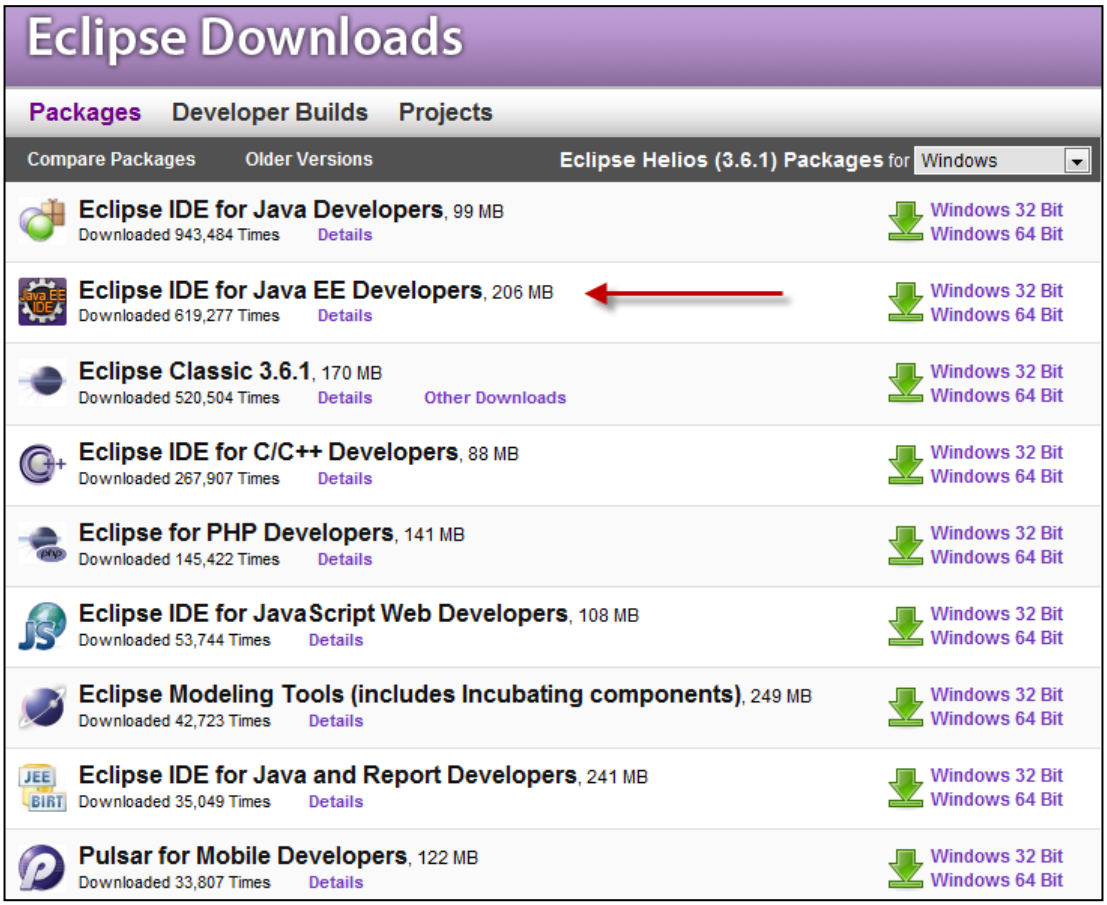


- Une fois le téléchargement terminé, décompressez le fichier téléchargé dans un dossier fixe, car il n'y a pas besoin de faire une autre manipulation pour l'installer.

- Gardez en mémoire le chemin d'accès au SDK car il faudra le rentrer dans Eclipse pour qu'il puisse interagir avec ce logiciel.

Installation du logiciel Eclipse Helios:

- Allez sur le site internet http://www.eclipse.org/downloads/ pour télécharger le logiciel de développement Eclipse Helios.



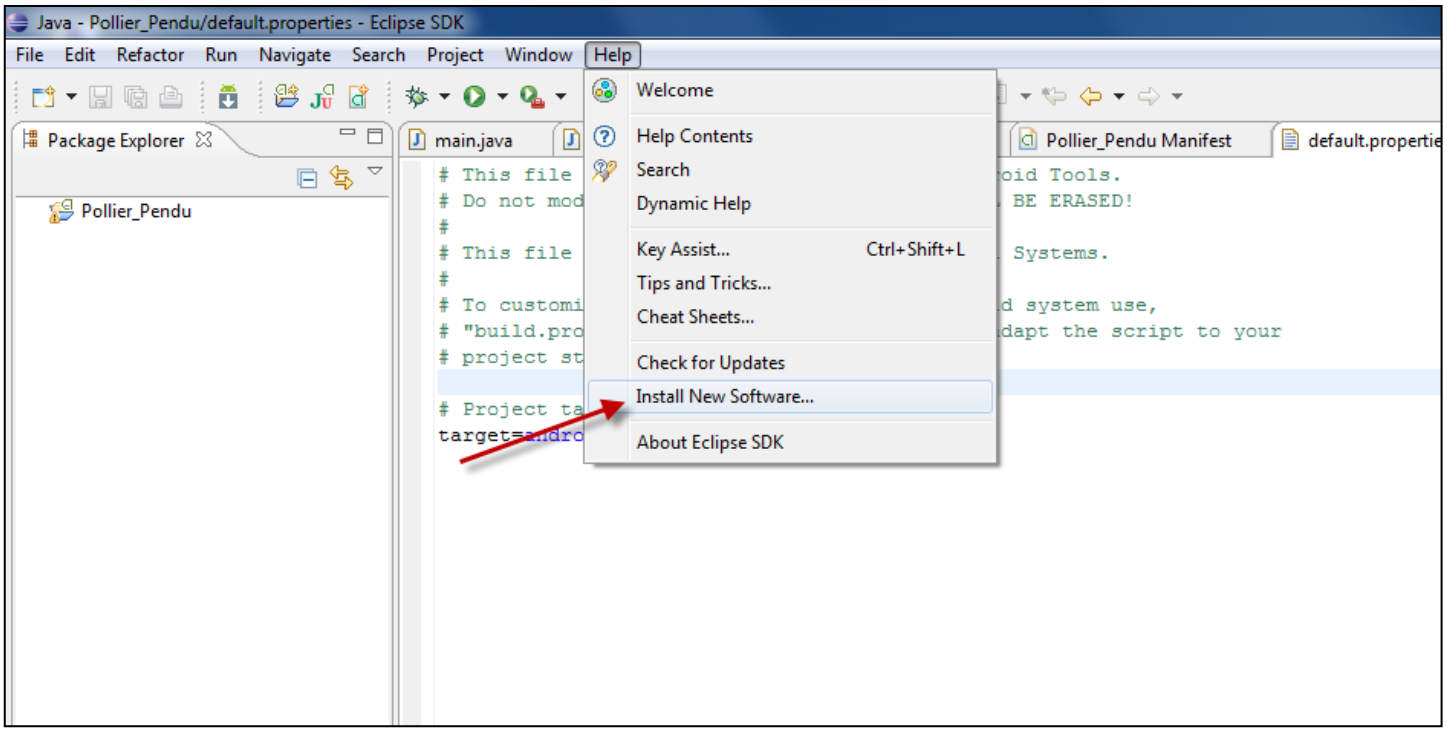
- Une fois téléchargé, lancez la décompression du fichier dans le dossier Programme files de votre ordinateur.

- Ensuite vous pouvez créer un raccourci du programme pour mettre sur le bureau en faisant click-droit sur le fichier «eclipse.exe» et créer raccourci.

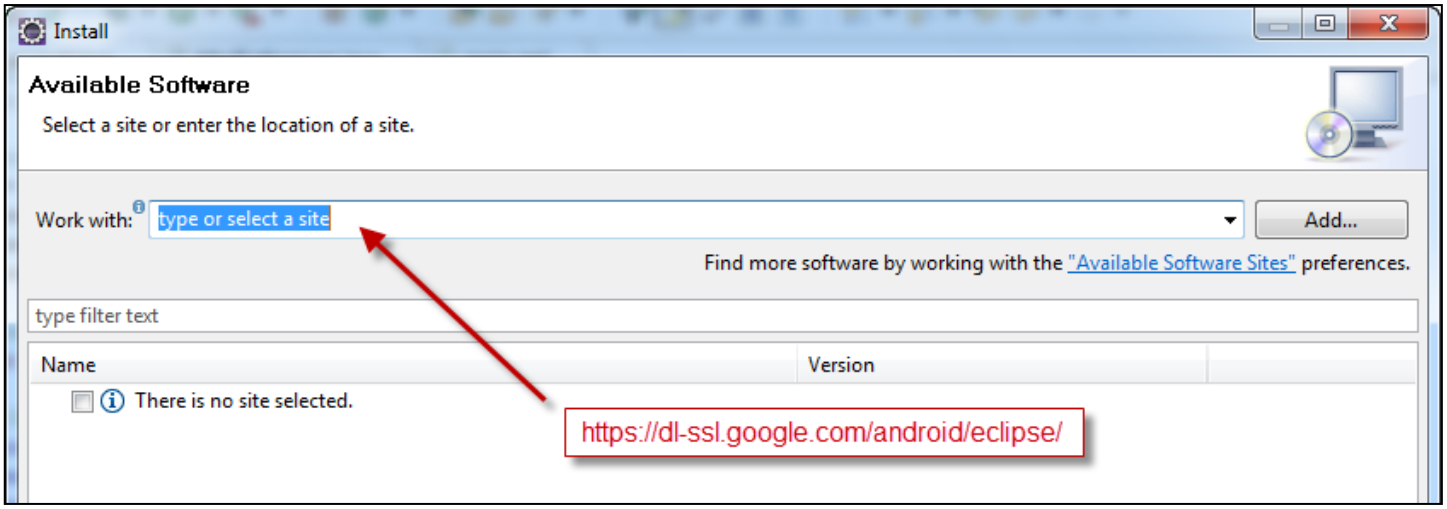
Installation du plugin ADT:

- Pour installer le plugin ADT, il faut ouvrir le programme Eclipse.

- Ensuite allez dans longlet «Help», Install New Software…»

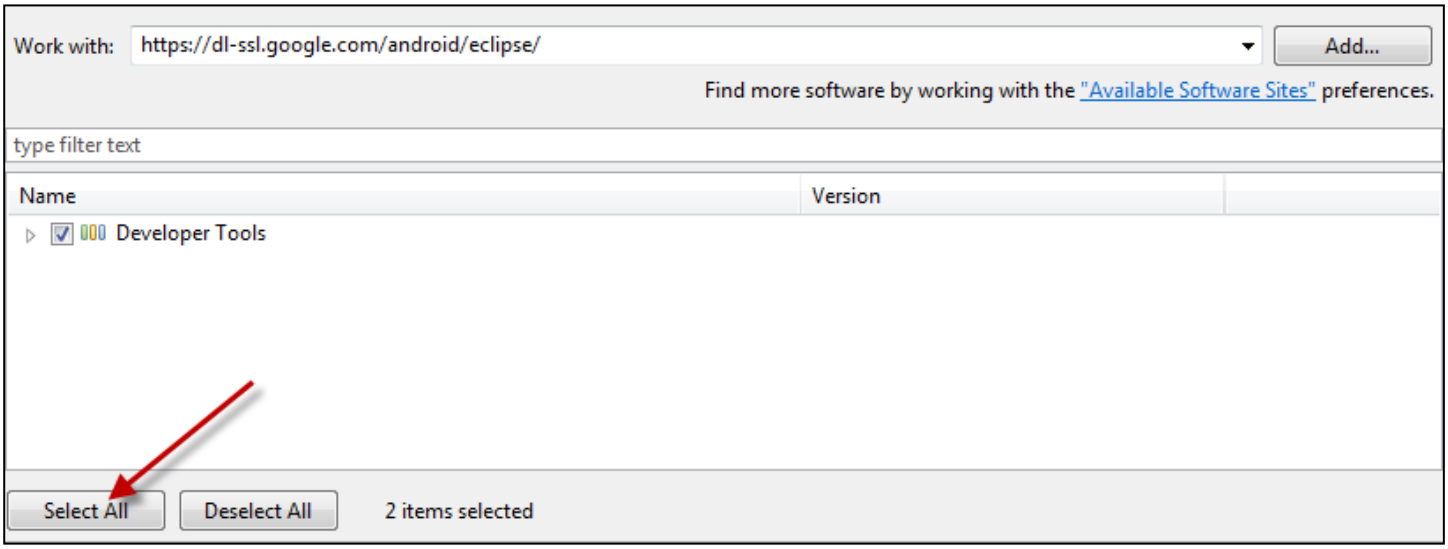


- Une fenêtre s'ouvre, il faut lui donner une adresse internet pour qu'elle puisse télécharger le plugin.

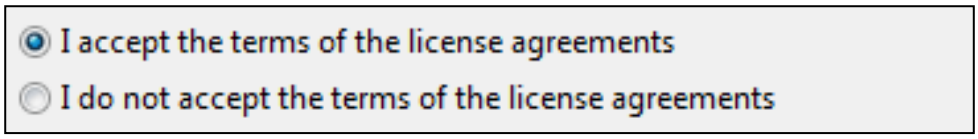


- L'adresse est: https://dl-ssl.google.com/android/eclipse/

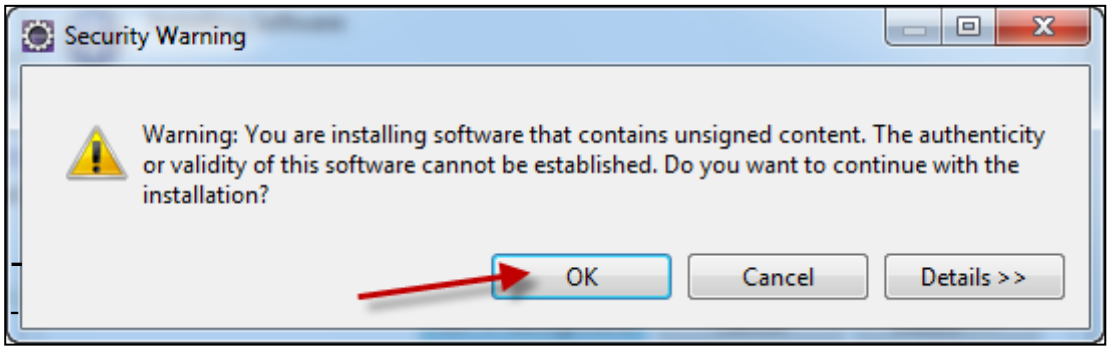
- Appuyez sur la touche «Select All» puis «Next»



- Encore une fois «Next» et vous arrivez sur une page ou il faut accepter les termes d'un contrat.



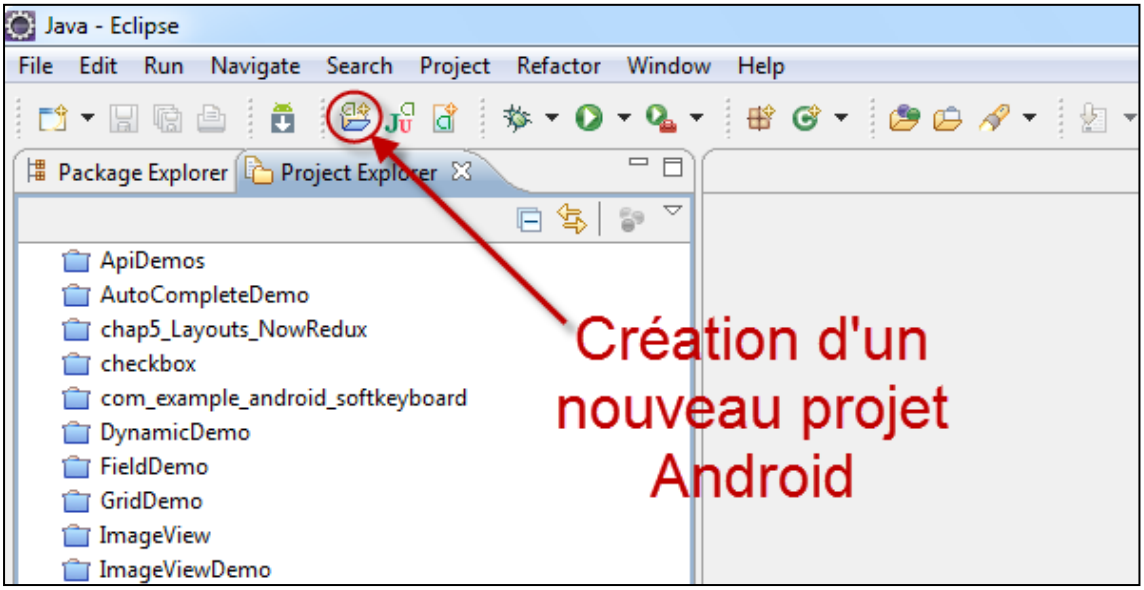
- Pour finir la procédure d'installation appuyer sur le bouton: «Finish»
- Une fenêtre de chargement s'ouvre et installe le plugin.
- Une fenêtre d'avertissement peut apparaître, appuyez sur «OK» pour continuer.



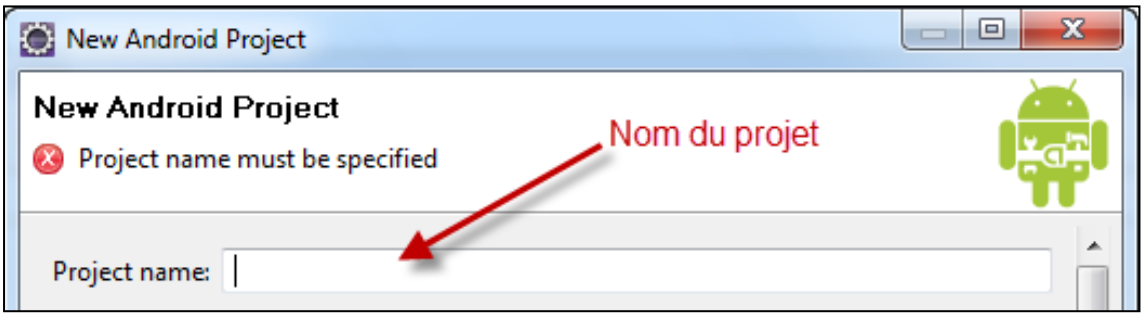
- Redémarrer Eclipse à la fin de l'installation pour que les changements soient effectifs.

Création d'un nouveau Projet

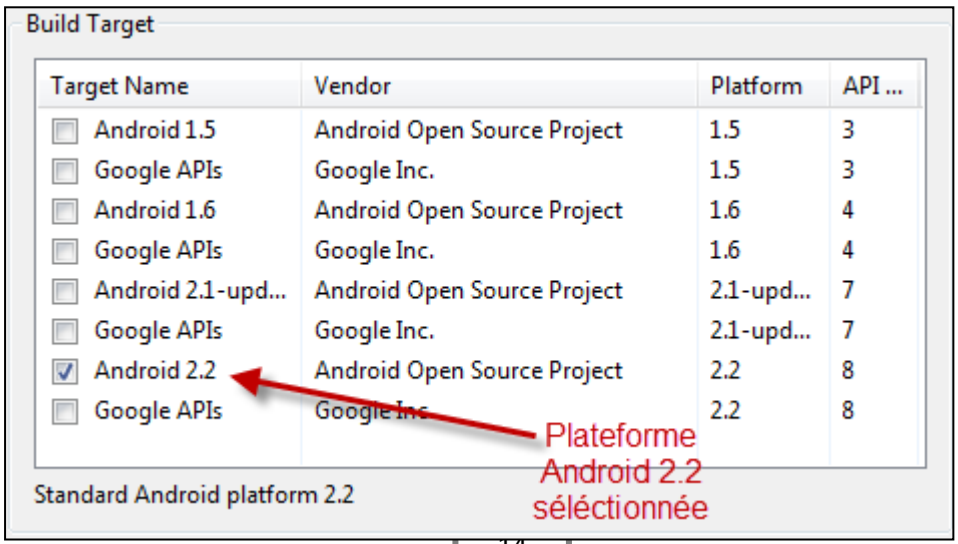
- Pour créer un nouveau projet, il faut ouvrir le programme Eclipse Hélios, puis appuyer sur l'icône de création de projet Android comme sur l'image ci-dessous.



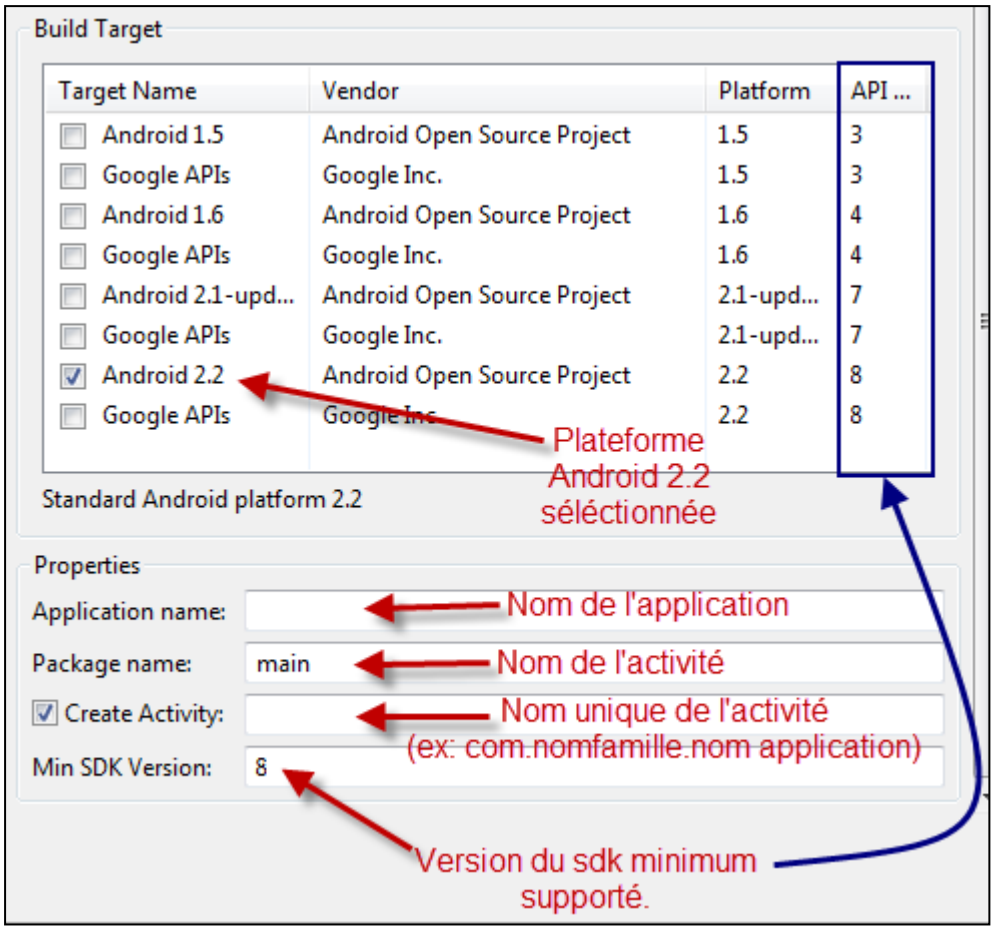
- Entrez le nom de votre projet



- Il faut alors sélectionner la plateforme sur laquelle votre application va fonctionner.



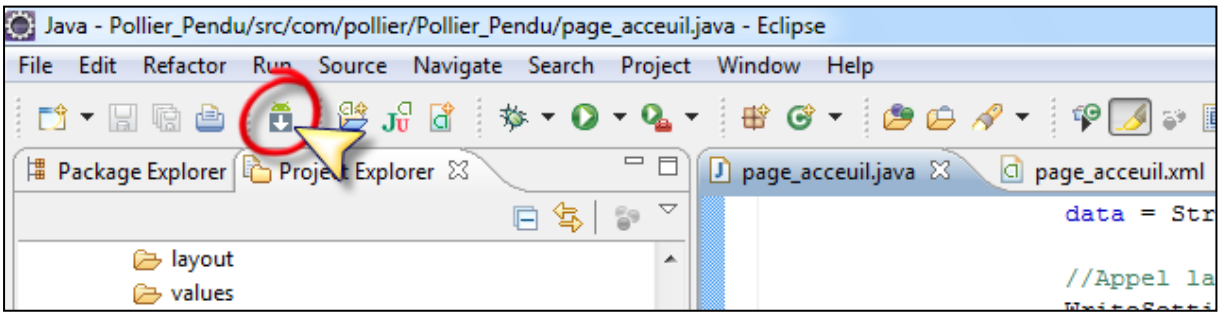
- Puis remplir les paramètres en suivant les indications de l'image ci-dessous.



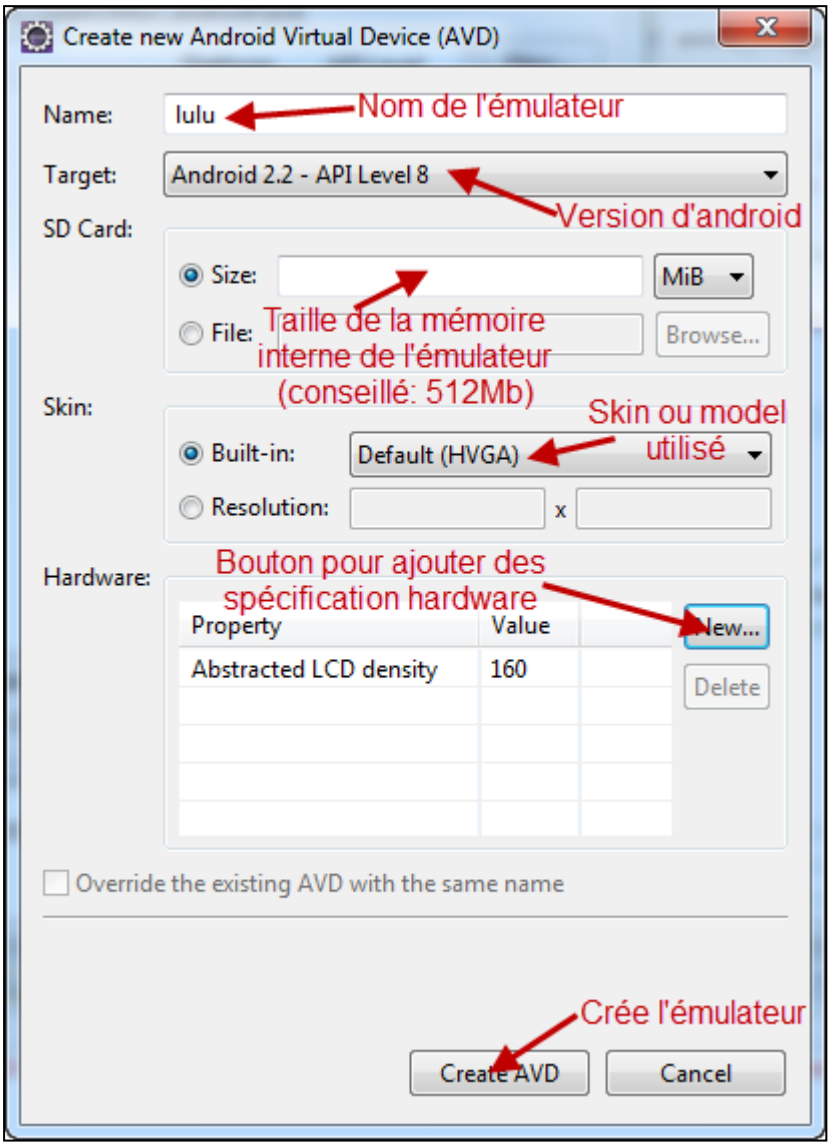
- Cliquez sur la touche «Finish» pour terminer la création du projet.

Création d'un émulateur

- Ouvrez Eclipse et appuyez sur l'icône (gestion des émulateurs) comme sur l'image ci-dessous.

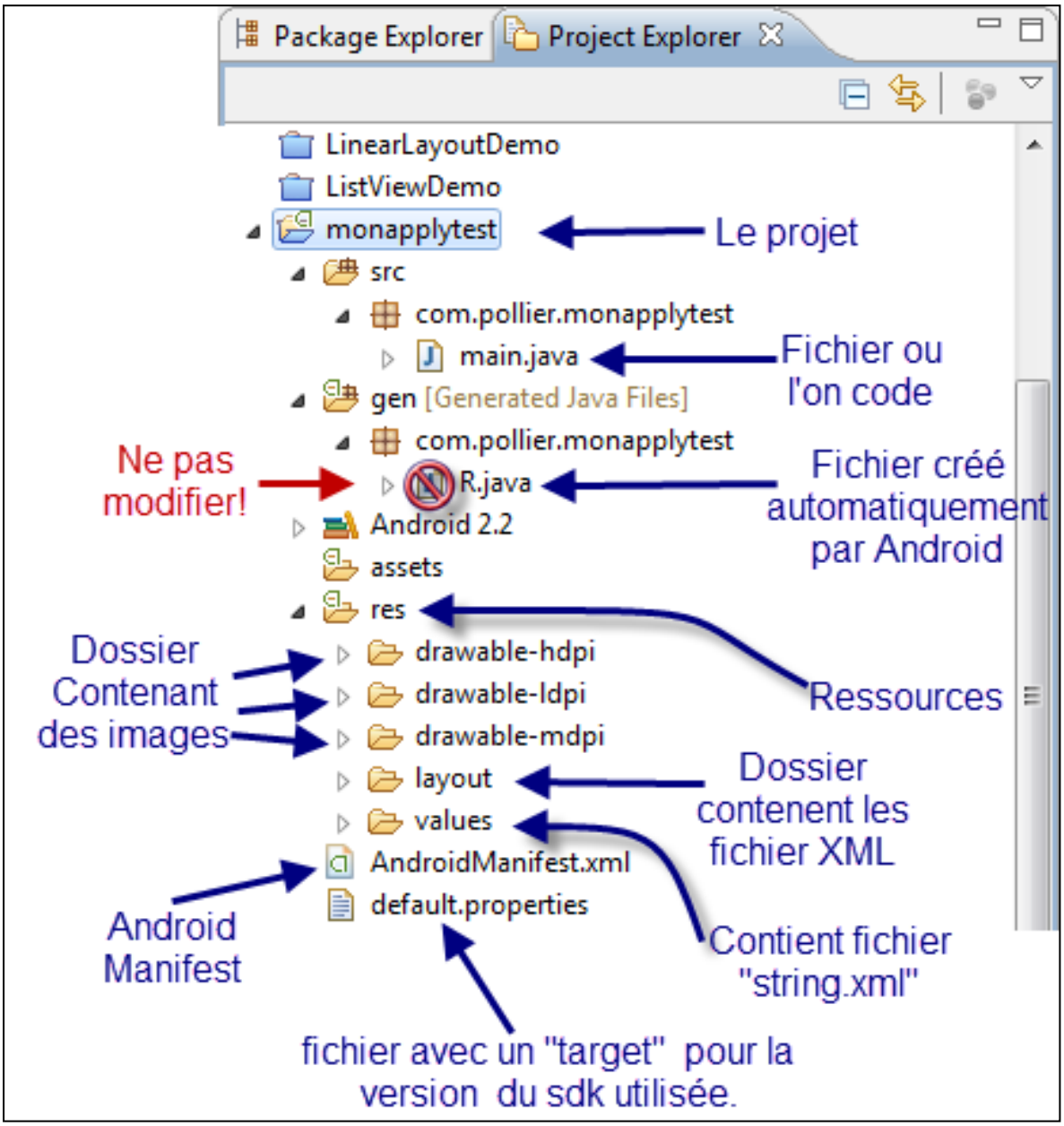


- Une fenêtre s'affiche, appuyer sur le bouton «New…» pour ouvrir l'éditeur de création.



- Une fois l'émulateur créé, vous pouvez le lancer en faisant un double-click dessus ou en appuyant sur «Start»

Arborescence d'un projet



Ressources

Physiquement, les ressources de l'application sont créées ou déposées dans le répertoire "res" du projet.
L'externalisation des ressources en permet une meilleure gestion ainsi qu'une maintenance plus aisée.
Le répertoire "res" sert de racine et contient une arborescence de dossier correspondant à différents types de ressources:

- res/values: (type de ressources: valeurs simples) Ce répertoire contient des fichiers (format XML) dont le nom reflète le type de ressources contenue ("arrays.xml" défini des tableaux).

- res/drawable: Fichiers .png, .jpeg qui sont convertis en bitmap.

- res/layout: Fichiers XML convertis en mises en page d'écrans (ou parties d'écrans), peuvent aussi être appelés gabarits.

- res/anim: Fichiers XML convertis en objets d'animations.

- res/xml: Fichiers XML pouvant être lus et convertis à l'exécution par la méthode "ressources.getXML".

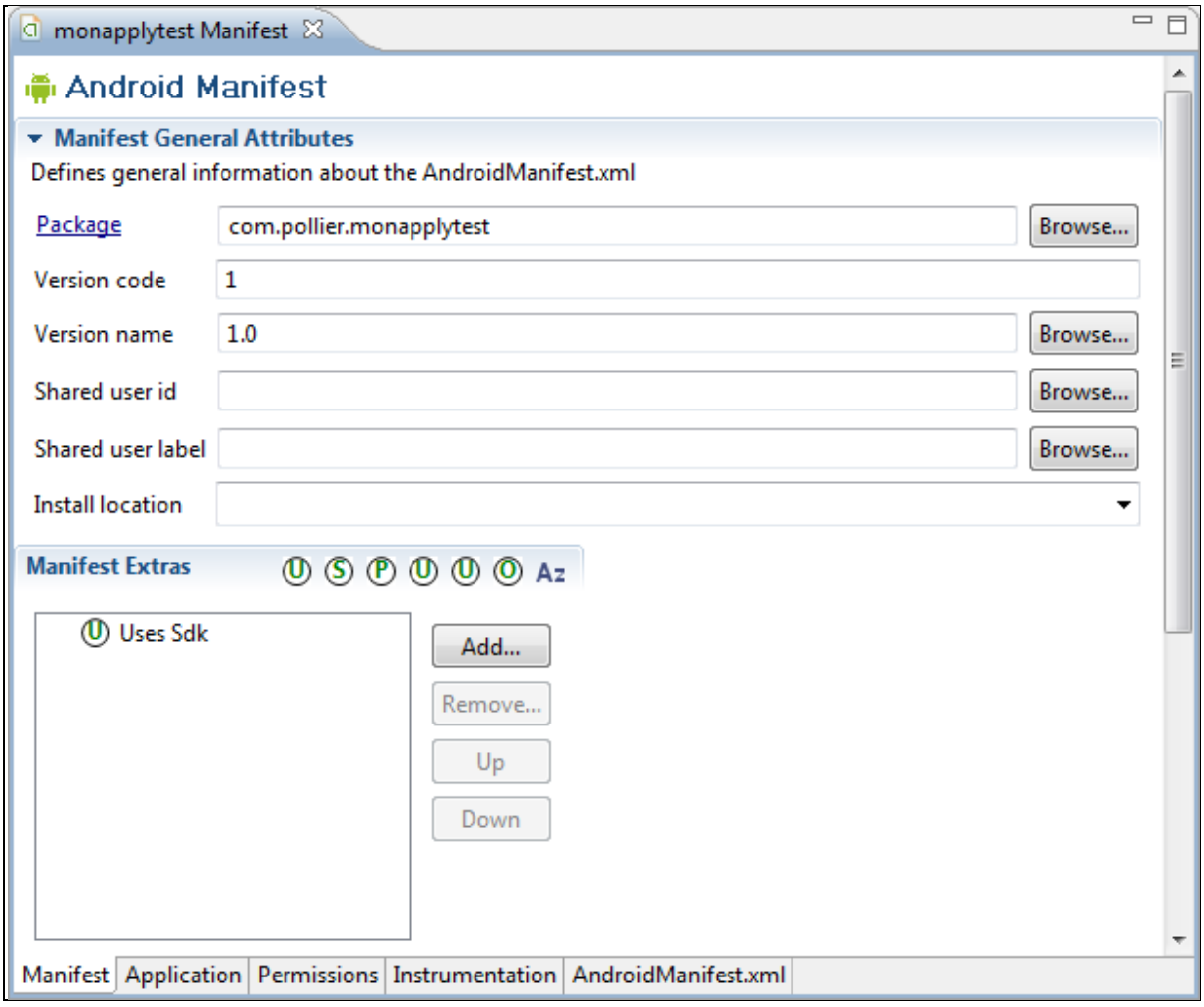
- res/raw: Fichiers à ajouter directement à l'application compressée et créée. Ils ne seront pas convertis.

Toutes les ressources sont placées dans un fichier de type APK qui constitue le programme distribuable de l'application.
Les ressources sont accessibles et utilisées depuis le code grâce à la classe statique "R". Cette classe est automatiquement générée en fonction des ressources présente dans le projet au moment de la compilation et de la construction de l'application.

Manifeste Android

Le manifeste est un fichier de type XML se trouvant à la racine de chaque projet. C'est dans ce fichier que l'on déclare ce que contiendra l'application: les activités, les services, les permissions, etc. On y indique aussi la façon dont ces composants sont relié au système Android en précisant, par exemple, l'activité principale qui doit apparaître dans le menu principal du terminal (Launcher).

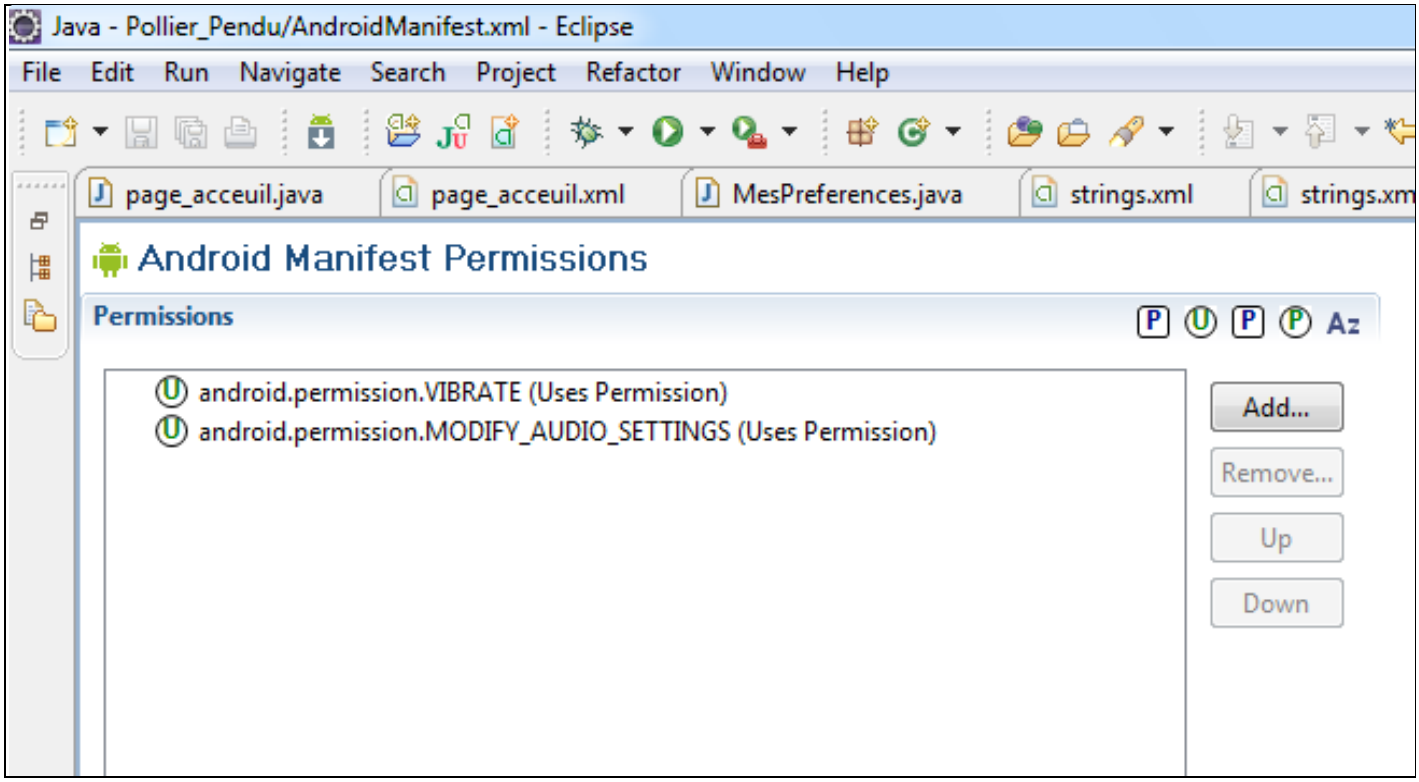
Fenêtre Eclipse de l'Android Manifeste



Exemple Code Manifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.pollier.monapplytest"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".main"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>
    <uses-sdk android:minSdkVersion="8" />

</manifest>
```

Fenêtre Permission Manifeste:



Exemples de codes:

1. **<uses-permission>**
   Les permissions qui seront déclarées ici seront utiles à l'application. Lors de l'installation de l'application on informera l'utilisateur que l'application utilise telle ou telle permission. Par exemple connexion réseau, vibreur, service audio… .

2. **<application>**
   Un manifeste ne contient qu'un seul nœud «application», mais plusieurs de type activités, services… .

3. **<activity>**
   Déclare une activité présentée à l'utilisateur. Si l'activité n'est pas inscrite, l'utilisateur ne la verra pas s'afficher.

4. **<service>**
   Déclare un composant de l'application en tant que service.

5. **<receiver>**
   Déclare un récepteur d'objet Intent. Cet élément permet à l'application de recevoir des objets alors qu'ils sont diffusés par d'autres applications ou le système.

6. **<provider>**
   Déclare un fournisseur de contenu qui permettra d'accéder aux données gérées par l'application.

7. **<uses-sdk>**
   Déclare quelle version du SDK Android utiliser pour ouvrir l'application.

Grace au plugin ADT, on a une interface visuelle pour gérer le manifeste.

Espace Développeur

Contenu d'un programme Android

Les composants principaux d'une application Android sont:

Les activités:

Ce sont les éléments de base de l'interface utilisateur. On peut considérer une activité comme une fenêtre ou une boîte de dialogue d'une application classique.

Les fournisseurs de contenus:

Les fournisseurs de contenus permettent un niveau d'abstraction pour toutes les données stockées sur le terminal et accessibles par les différentes applications.

Le développement Android pousse aux partages de données entre applications, le fournisseur de contenu permet l'échange d'information en gardant le contrôle sur la façon dont on accédera aux données.

Les services:

Les activités et les fournisseurs de contenu ont une durée de vie limitée et peuvent être éteints à tout moment. Les services, par contre, ne sont pas prévus pour être arrêtés, et fonctionnent même si nécessaire indépendamment de toute activité.

Exemple: On pourrait utiliser un service pour vérifier les mises à jour d'un flux RSS même si l'activité n'est plus en cours d'exécution.

Les intentions (Intents):

Les Intents forment un mécanisme sophistiqué et complet permettant aux activités et aux applications de communiquer et d'interagir entre elles.

Classe Intent:

La communication du système interne Android est basée sur l'envoi et la réception de messages exprimant l'intention d'effectuer une action. Chacun de ses messages peut-être émis à destination d'un autre composant de l'application (ex: un service) ou celui d'une autre application.

Issu de la classe Intent, ce message permet donc de véhiculer toutes les informations nécessaires à la réalisation de l'action:

- Informations à destination du composant qui le réceptionnera (action à effectuer et les données avec lesquelles agir).

- Informations nécessaires au système pour son traitement (catégorie du composant ciblé par le message et instructions d'exécution de l'action).

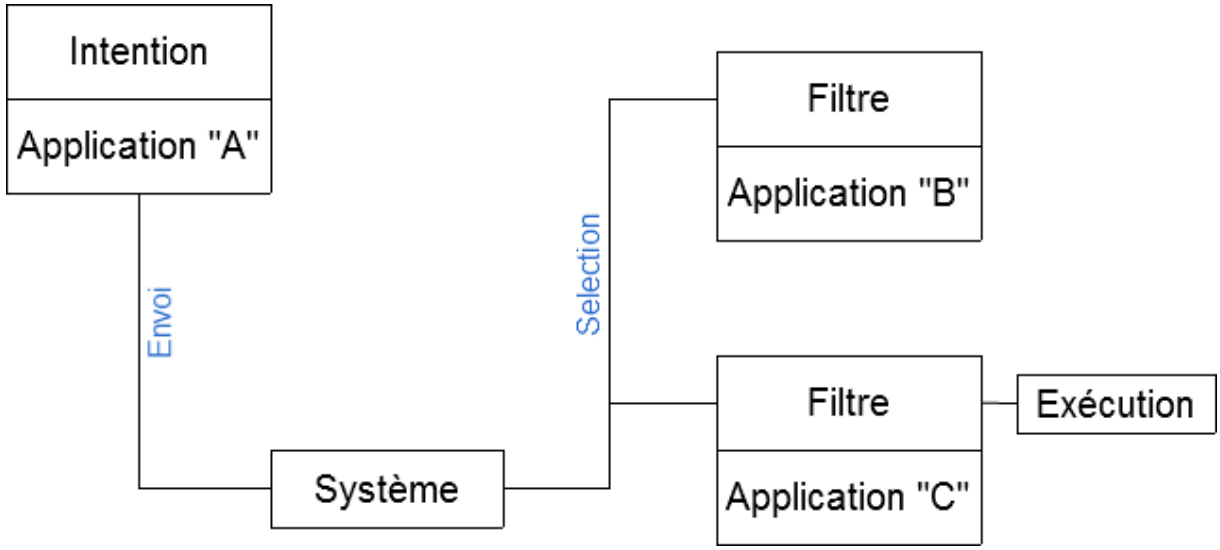
On peut envoyer des Intents au système de deux façons différentes:

- soit en ciblant un composant précis d'une application (mode explicite)

- soit en laissant le système déléguer le traitement de cette demande au composant le plus approprié (mode implicite).

Un système de filtre permet à chaque application de filtrer et de gérer uniquement les Intents qui sont pertinents pour elle. C'est ainsi qu'une application peut être en état d'inactivité mais tout en restant à l'écoute des Intents du système

Schéma Intents et Filtres:



Objets Intents

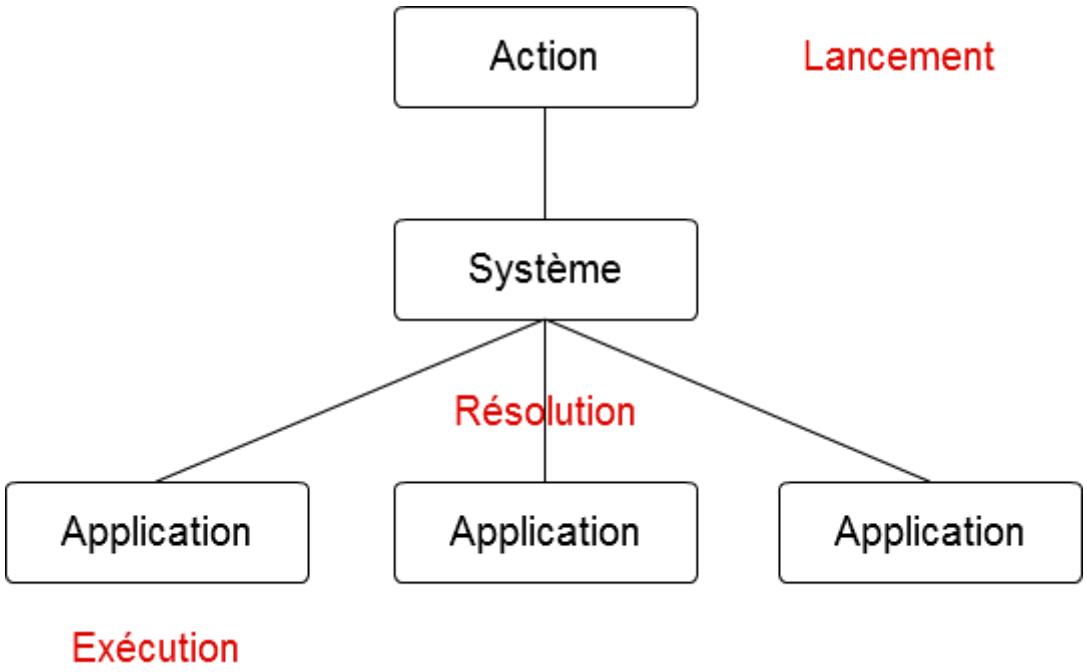
Un objet Intent véhicule toutes les informations nécessaires à la réalisation d'une action (ou à la réception d'information).

Pour décider du composant le plus approprié à utiliser, le système se base sur les informations que l'on spécifie dans un objet Intent.

- Le nom du composant ciblé permet de spécifier de façon non ambiguë le nom du composant qui sera utilisé pour réaliser l'opération. (Si le nom n'est pas spécifié, le système déterminera le composant le plus approprié.)
- L'action: chaines de caractères définissant l'action à réaliser. (Dans le cadre d'un récepteur d'Intents, il s'agira de l'action qui s'est produite et pour laquelle le système ou l'application informe toutes les autres.)
- Les données: le type de contenu (MIME[4]) sous la forme d'une chaine de caractères et le contenu ciblé sous la forme d'un URI[5].
- La catégorie: cette information permet de cibler plus précisément qui devra gérer l'Intent émis. (par exemple, l'utilisation de la constante "CATEGORY_BROWSABLE" demande le traitement par un navigateur web.
- Les drapeaux: principalement utilisés pour spécifier comment le système doit démarrer une application.

Ne pas oublier d'ajouter des permissions (privilège spéciaux) dans le manifeste pour permettre à certains Intents de bien aboutir.

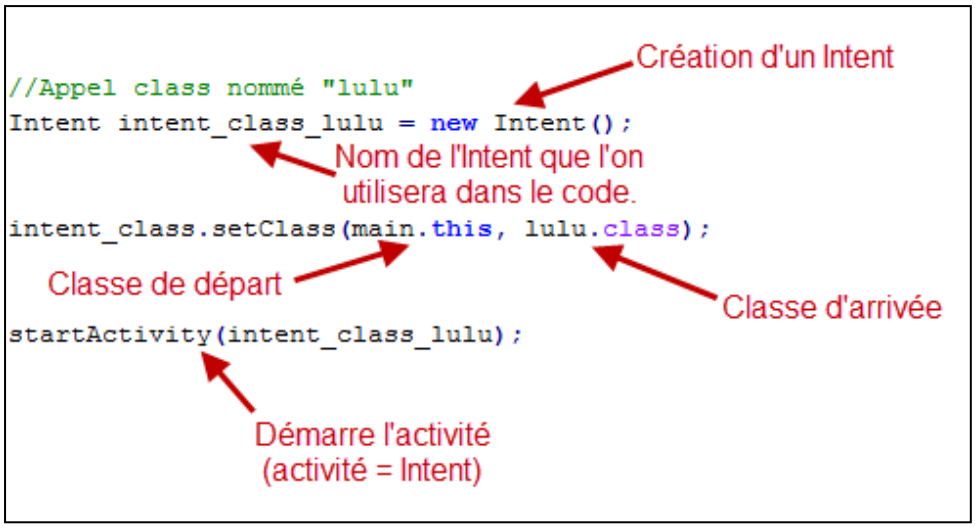
Schéma traitement Intents:



[4] MIME: Le type MIME (Multipurpose Internet Mail Extensions) permet de savoir de quelle manière afficher le document.

[5] URI: Un URI (Uniform Ressource Identifier) est un identifiant unique permettant d'identifier une ressource de façon non ambiguë sur un réseau. (exemple URI: "Schéma://hote:port/chemin",Schéma = type)

Ouverture d'une autre classe via Intent:



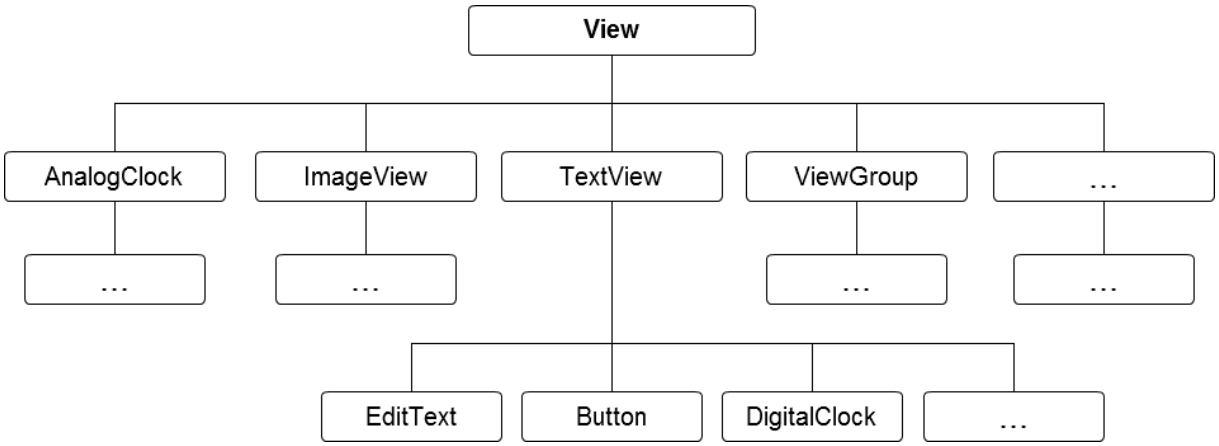
Classe View

Les vues permettent la construction de l'interface d'une application Android. Ces objets représentent des éléments qui s'afficheront à l'écran et qui permettront à l'application d'interagir avec l'utilisateur via des événements.

Chaque écran Android contient un arbre d'éléments de type View dont chaque élément est différent des autres de par sa forme, sa taille, sa fonctionnalité…

La plupart des éléments que nous utilisons (bouton, textes…) sont fournis par la plate-forme Android, mais il nous reste toujours la possibilité de créer des éléments personnalisés en partant du code d'un objet déjà existant.

D'un point de vue plus technique, le mot widget désigne l'ensemble des vues standards incluses dans la plate-forme. Elles font parties du paquetage «android.widget». Les vues héritant toutes de la classe View, chaque widget hérite aussi de cette classe. Ainsi l'élément «Button» hérite de «TextView», qui hérite de la classe View.



Il existe deux façons de décrire une interface:

- soit via une définition XML
- soit depuis le code (Java)

Le fait de construire une interface au sein du code d'une activité permet par exemple de créer des interfaces dynamiquement.

Instanciation d'un widget au sein du code d'une activité:

```
Button button = new Button(this);
button.setText("Cliquez  ici");
button.setOnClickListener(this);
```

Définition XML:

```
<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Cliquez ici"
></Button>
```

Utilisation de Widgets

Les widgets sont des composants qui serviront à l'utilisateur pour interagir avec l'application. Voici quelques exemples de widgets que vous seriez amené à utiliser un jour:

- **AnalogClock**
  Affiche une horloge à aiguilles dans l'application.
- **AutoCompleteTextView**
  Ce widget est une sorte d'hybride d'EditText et de Spinner.
  Le texte saisi par l'utilisateur est traité comme un préfixe de filtrage: il est comparé à une liste de préfixes candidats et les différentes correspondances sont affichée dans une liste de choix qui ressemble à un Spinner.
- **Button**
  Le widget «Button» est une sous-classe du widget «TextView».
- **CheckBox**
  Hérite du widget «CompounButton». Propose un choix élémentaire à l'utilisateur «oui» ou «non».Pour récupérer à tout moment l'état de la case à cocher, on utilise la méthode «isChecked».

- **Chronometer**
  Permet de mémoriser le temps écoulé depuis un point de départ. Il suffit d'utiliser les methode «start()» quand on veux démarrer et «stop()» quand on veux s'arrêter.
- **ColorStateListe**
  Permet de préciser plusieurs couleurs correspondant à différentes situations.
- **CompounButton**
  Hérite du widget «Button». Un bouton à deux états, coché et non-coché. Lorsque le bouton est pressé ou cliqué, l'état change automatiquement.
- **DatePickerDialog**
  Affiche des boutons permettant de saisir une date.
- **DatePicker**
  Affiche des boutons permettant de saisir une date. Il s'adapte à la région et la langue du téléphone. Attention, si vous devez l'utiliser, les mois commencent à 0 (0= Janvier)
- **DigitalClock**
  Affiche une horloge digitale dans l'application (hh:mm:ss).
- **EditText**
  Boite d'édition permettant de saisir du texte.
- **ImageButton**
  Une sous-classe du widget «ImageView», il fonctionne comme le widget «Button», sauf que l'on peut lui définir une image en changeant l'attribut «android:src» comme une ImageView.
- **ImageView**
  Affiche une Image. L'attribut «android:src» permet de lui préciser l'image à utiliser. L'image vient en général des ressources graphiques (res/Drawable).
- **ListView**
  Conteneur permettant d'afficher des données sous forme de liste.
- **ProgressBar**
  Affiche une barre de progression ou une animation. Widget de sortie dont le but consiste à indiquer l'état d'une progression à l'utilisateur.
- **RadioButton**
  Dérive du widget CompounButton. Propose de faire un choix de réponse unique à peu près comme le widget CheckBox. Bouton à deux états pour une utilisation en groupe.
- **RadioGroup**
  Le fait de mettre les RadioButton dans un RadioGroup permet de faire qu'il n'y en ait toujours qu'un seul de coché à la fois (Basculer d'un RadioButton à l'autre).
- **RatingBar**
  Affiche une barre comportant des étoiles pour représenter ou récupérer une notation de l'utilisateur. Récupérer la note grâce à la méthode «getRating».

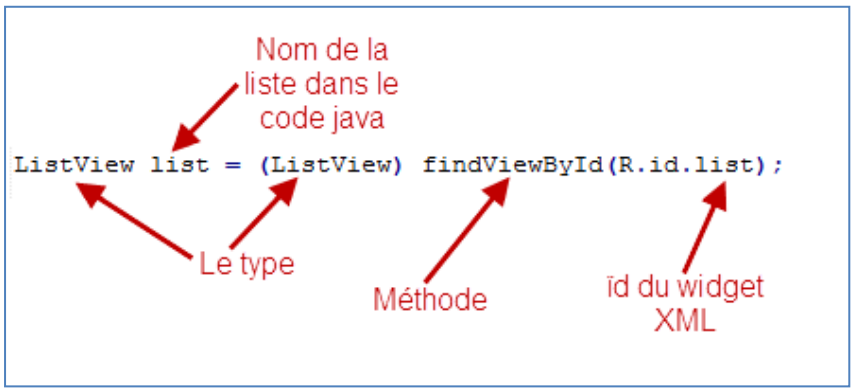
- **SeekBar**
  Ce widget hérite du widget «ProgressBar». Mais au lieu d'être un widget de sortie c'est un widget d'entrée. C'est à dire qu'il permet à l'utilisateur de saisir une valeur prise parmi un intervalle donné (exemple: contrôle du volume).
- **Spinner**
  C'est une boîte déroulante. Une boite déroulante est un mix entre une liste et un AlertDialog (fenêtre qui s'ouvre pour prévenir l'utilisateur d'un événement.).
- **TextView**
  Affiche un texte. On utilise la méthode «setText()» pour lui ajouter du texte.
- **TimePickerDialog**
  Affiche des boutons pour régler une heure. Elle est sous la forme hh:mm on peut aussi indiquer si elle fonctionne sur 12heures avec l'indication AM/PM ou 24heures.
- **TimePicker**
  Affiche des boutons pour régler une heure. Elle est sous la forme hh:mm on peut aussi indiquer si elle fonctionne sur 12heures avec l'indication AM/PM ou 24heures.

Atteindre un widget créé dans un fichier XML

Grace à la méthode «findViewById ()» on peut accéder à nos widgets depuis le code java, en lui passant l'identifiant numérique (id) du widget concerné.

L'identifiant a été créé par Android dans la class «R» et est de la forme «R.id.id_widget_xml».

Exemple appel widget:



Listener

Un listener en français se traduit par un écouteur. Un écouteur, comme son nom l'indique, écoute une source jusqu'à ce qu'elle soit utilisée.

Pour qu'une action soit réalisée, il faut qu'un émetteur envoie un signal dans le canal d'écoute pour que le listener se déclenche.

Action: Clic sur un bouton

Listener: Ecoute si le bouton est cliqué

Code:

```
monBouton.setOnClickListener(new OnClickListener()
{
    @Override
    public void onClick(View v)
    {
    // TODO Auto-generated method stub
    }
} );
```

Si vous avez envie d'écouter le changement d'état d'un widget, alors n'hésitez pas à chercher une méthode utilisant un Listener.

Animations

Nous pouvons animer une View directement depuis le code ou depuis un fichier XML.

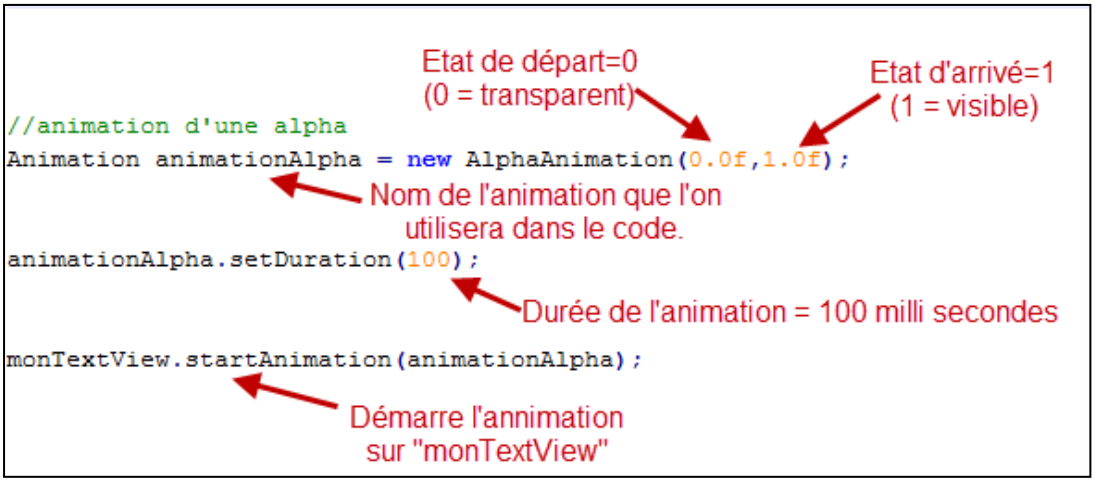
Différentes Animations:

- Opacité (**Alpha**): permet de jouer sur la transparence ou l'opacité d'une vue.
- Échelle (**Scale**): permet de spécifier l'agrandissement/réduction sur les axes X et Y à appliquer à la vue.
- Translation (**Translate**): permet de spécifier la translation/déplacement à effectuer par la vue.
- Rotation (**Rotate**): permet d'effectuer une rotation selon un angle en degré et un point de pivot.

Chaque type d'animation possède une sous-classe dérivant de «Animation»:

- AlphaAnimation
- ScaleAnimation
- TranslateAnimation
- RotateAnimation

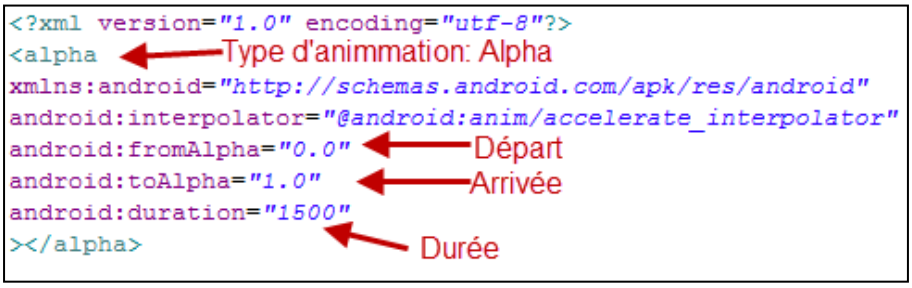
Animer par le code:



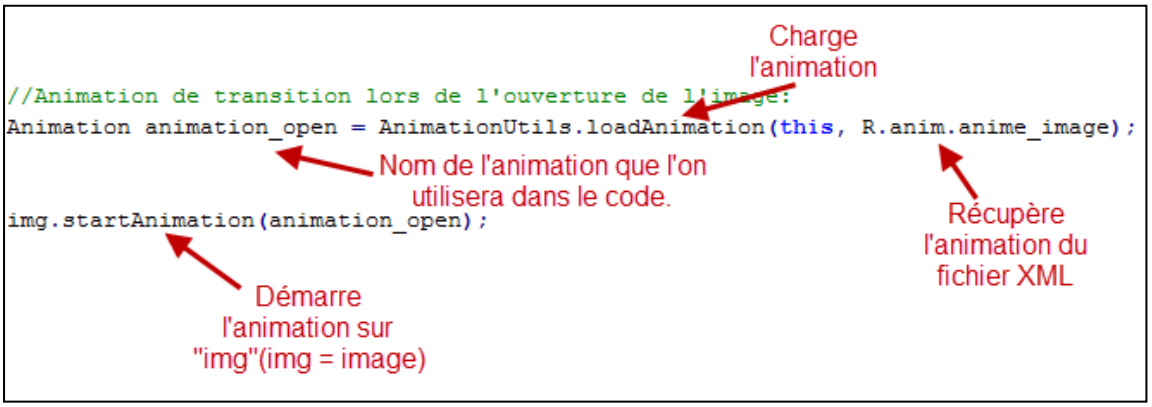
Animer par fichier XML:

Il faut créer un dossier «res/anim» et y mettre un fichier xml avec à l'intérieur les paramètres de l'animation.

Exemple pour une image cette fois:



Code pour utiliser l'animation:



Interface d'une application

Nous avons vu précédemment qu'un widget pouvait être créé de deux manières différentes directement en code Java ou depuis un fichier XML. L'avantage d'utiliser un fichier XML, c'est qu'il est plus facile à lire et à créer et nous permet aussi de créer des interfaces multilingues. Grace au plugin ADT nous avons en plus la possibilité de prévisualiser ce que représente le code d'un fichier XML, un grand avantage quand on n'est pas encore un As pour la création d'interfaces.

Vous trouverez ci-dessous les outils pour créer une interface d'application depuis un fichier XML.

Conteneurs

Les conteneurs sont indispensables, ils permettent de disposer un ensemble de widgets pour obtenir la présentation voulue d'une interface.

- **AbsoluteLayout**
  Le contenu d'un AbsoluteLayout est disposé en fonction de coordonnées spécifiques. On lui indique où placer un fils en lui précisant ses coordonnées X et Y. Puis Android le positionne à cet endroit sans poser de question.
  Ceci a l'avantage de fournir la possibilité de faire un positionnement précis mais malheureusement, l'application ne sera utilisée que sur une certaine taille d'écran. Ce conteneur n'est plus très utilisé, et même déconseillé aujourd'hui.

- **FrameLayout**
  Le FrameLayout est le conteneur de contenu des onglets. Chaque contenu des onglets est un fils du FrameLayout.

- **HorizontalScrollView**
  Fonctionne comme le conteneur «ScrollView», mais de manière horizontale. Notez qu'il faut choisir soit un scroll vertical, soit un scroll horizontal. On ne pourra pas intégrer les 2 en même temps.

- **LinearLayout**
  LinearLayout est un modèle reposant sur des boîtes. Les widgets ou le conteneur fils sont alignés en colonne ou en lignes, les uns après les autres.
  Pour configurer un LinearLayout on peut agir sur 5 paramètres tels que l'orientation, le modèle de remplissage, le poids, la gravité et le remplissage(Padding).

- **RelativeLayout**
  Il place les widgets relativement aux autres widgets du conteneur et de son conteneur parent. Vous pouvez donc placer votre widget par rapport à un autre widget, soit en dessous ou en dessus, soit à gauche ou à droite.

- **ScrollView**
  ScrollView est un conteneur qui fournit un défilement vertical à son contenu.

- **ViewFlipper**
  Hérite de FrameLayout, permet de passer d'un onglet à un autre sans cliquer sur l'onglet mais en faisant un déplacement avec le doigt sur l'écran.

Exemple d'Interface:

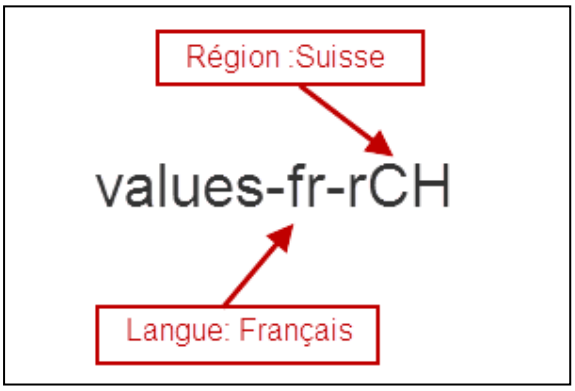
Code relatif à l'exemple d'interface:

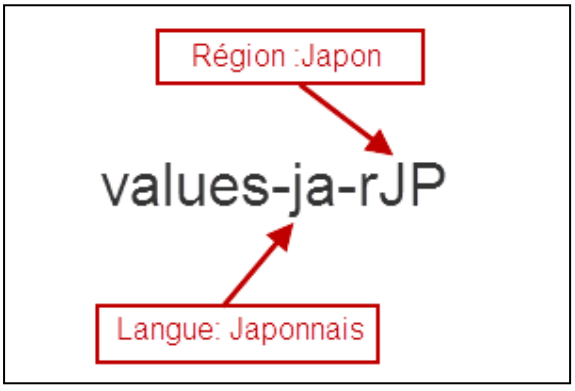
```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:id="@+id/RelativeLayout01"
android:layout_width="wrap_content"
android:layout_height="wrap_content">

<TextView android:layout_height="wrap_content"
android:id="@+id/tv_titre"
android:layout_width="fill_parent"
android:gravity="center_vertical|center_horizontal"
android:text="Jeu du pendu !!"
android:layout_centerHorizontal="true"
android:layout_alignParentTop="true"
></TextView>

<ImageView android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/image_acceuil"
android:layout_centerHorizontal="true"
android:layout_below="@+id/tv_titre"
android:background="@drawable/image_titre"
></ImageView>

<RadioGroup android:layout_width="wrap_content"
android:layout_marginRight="5dp"
android:layout_alignParentBottom="true"
android:id="@+id/RadioGroup01"
android:layout_alignParentLeft="true"
android:layout_height="150dp">

    <RadioButton android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/rb_debutant"
    android:text="Débutant"
    android:checked="true"
    ></RadioButton>
    <RadioButton android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/rb_modere"
    android:text="Modéré"
    ></RadioButton>
    <RadioButton android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/rb_expert"
    android:text="Expert"
    ></RadioButton>
</RadioGroup>

<Button android:text="Play"
android:textSize="40px"
android:textStyle="bold"
android:id="@+id/demarrer"
android:layout_alignParentBottom="true"
android:layout_alignParentRight="true"
android:layout_width="wrap_content"
android:layout_toRightOf="@+id/RadioGroup01"
android:layout_height="150dp"
></Button>

</RelativeLayout>
```

Internationalisation d'une application

Si un jour vous comptez mettre en vente votre application sur l'Android Market, il faut prendre en compte que la majorité des utilisateurs parleront anglais. Dans l'idéal il faut que votre application change de langue en fonction de la langue du Smartphone.
Supposons que vous voulez fournir des textes anglais et français. Pour une application non traduite, vous placeriez normalement ces chaînes de caractère dans le fichier «res/values/strings.xml» mais, pour pouvoir reconnaître à la fois l'anglais et le français, vous devez créer deux répertoires. Celui de base sera pris automatiquement si la langue du téléphone est en anglais. Ensuite, il faut créer le deuxième, «res/ values-fr-rCH/strings.xml» est pour la langue française et région suisse.

Exemples:





On peut faire le même procédé avec les images ou toutes sortes de ressources du projet.

Liens avec listes des langues et pays:

- www.loc.gov/standards/iso639-2/php/code_list.php

- www.iso.org/iso/country_codes/iso_3166_code_lists/english_country_names_and_code_elements.htm

Conclusion:

Il reste encore un nombre d'éléments important à découvrir concernant le développement d'applications, j'espère toutefois que se document vous a aidé à progresser.

Bonne continuation sur la voie du développement d'applications Android !